基于调度历史数据在线预测作业执行时间 *

许伦凡，熊　敏，肖永浩†
(中国工程物理研究院 计算机应用研究所，四川 绵阳 621900)

摘　要：在基于回填策略的调度系统中，作业的预估执行时间是不可缺少的参数。传统基于用户预估的执行时间通常准确性较差。结合分类和基于实例的学习方法，综合使用模板相似和数值相似方法，在历史调度数据中获取当前作业的相似作业，并使用其历史信息预测当前作业执行时间。使用调度历史中的用户名、分组名、队列名、应用名、用户请求处理器数、用户请求(预估)执行时间和用户请求内存量等属性进行训练和预测，算法中涉及的参数使用遗传算法确定。数值实验表明，相对于文献[1]，在使用更少参数的前提下，得到了与文献结果中相近的低估率，并获得了更低的平均绝对误差。在 HPC2N04 和 HPC2N05 日志数据集上，平均绝对误差分别降低了 43%和 77%。研究了使用在线预测替换用户估计对作业调度的影响，对结果进行了初步分析，并指出了今后的改进方向。
关键词：执行时间预测；作业调度；遗传算法；K-近邻
中图分类号：TP181　　doi: 10.3969/j.issn.1001-3695.2018.08.0624

On-line prediction of application using schedule historical data

Xu Lunfan, Xiong Ming, Xiao Yonghao+
(*Institute of Computer Application, China Academy of Engineering Physics, Mianyang Sichuan* 621900, *China*)

Abstract: In the scheduling system based on the backfilling strategy, the estimated application runtimes is an indispensable parameter. Traditional runtimes based on user estimating is usually less accurate. In this paper, a combination of the categorization and the instance based on learning methods defined the job similarity. This paper used the template similarity and numerical similarity method to find the similar jobs of the future one, and used historical data to predict the runtimes. The proposed algorithm only take seven job attributes into account. The attributes include “user name”, “group name”, “queue name”, “application name”, “requested number of processors”, “requested runtime”, “requested memory”. It applied Genetic Algorithm to train the best parameters, and used similar jobs attributes to predict runtimes. Compared with the literature [1], experimental results show that the predictor achieved a similar underestimate rate on the premise of using fewer parameters. Moreover, the proposed predictor gets a lower mean absolute error. On the HPC2N04 and HPC2N05 datasets, the mean absolute errors reduced 43% and 77% respectively. This paper studied the effecting of using online prediction to replace user estimation on job scheduling, preliminary analyzed the results and pointed out the future improvement directions.
Key words: application runtimes prediction; job scheduling; genetic algorithm; K-nearest neighbor

0　引言

基于回填(backfilling) [4]的作业调度是当前超算中心的核心调度算法之一。传统回填算法是在先来先服务(FCFS)的基础上将小作业回填到空闲 CPU，以提高 CPU 利用率。算法是以队列作业所需要的CPU数量为条件，对于满足条件的作业又需要以待回填作业的执行时间为衡量标准，只有同时满足这两个条件的作业才能够回填，这就需要估计每个作业的执行时间，即作业预估执行时间。

一般调度系统中作业预估执行时间由用户提供，从文献[4,5]中的研究表明，用户提供的作业预估时间与作业真实执行时间有很大的误差。Chiang 等人[10,11]指出，在使用高性能回填策略时，预测的准确性会对调度性能产生较大影响，此即意味着更准确地执行时间估计可以显著提高调度系统的性能[10]。文献[16]研究了在网格环境下让用户在提交作业之前精确的预测作业执行时间。

另一种获得作业预估时间的方法是由系统自动生成预估时间。例如文献[1,3,6,8,9,13,15]中基于历史数据挖掘的预测方法，此类方法相对于文献[6,7]等提出的预测方法更容易操作。Tsafrir 等人[6]提出的朴素预测算法生成的时间相对用户直接给出的估计，其准确程度有显著提升。文献[8]针对用户提交作业的行为模式进行聚类，与已有的预测算法相比预测精度可以提高 5.6%，同时将计算花费的时间减少到 3.8%。文献[17]提出一种基于自组织映射的运行时间预测方法，该方法预测精度优于基于实例的学习和统计平滑。Tran 等人[1]提出的模板预测方法是当前最先进的算法，在降低作业低估率的同时，相对于文献[6]的平均绝对误差(MAE)改进了 20%以上。

目前已有的工作[1,6,8,9,13]都是通过对作业日志的离线分析得到预测的执行时间，本文将基于历史数据在线预测作业执行时间。

收稿日期：2018-08-17；**修回日期：**2018-10-10　**基金项目：**国家重点研发计划资助项目（2016YFB0201504）
作者简介：许伦凡（1994-），男，湖南衡阳人，硕士研究生，主要研究方向为并行计算、智能调度等；熊敏（1980-），女，高级工程师，博士，主要研究方向为计算几何、微分几何、机器学习；肖永浩（1981-），男（通信作者），博士，高级工程师，主要研究方向为高性能计算、作业调度、机器学习（xiao_yonghao@163.com）.

1 相关工作

基于调度历史的作业执行时间预测研究是以超算系统中用户会多次执行相同的应用程序进行计算为背景，在此背景下，以"相似的作业拥有相似的执行时间"为假设前提，通过已经完成作业的执行时间来预测相似作业的执行时间。从寻找相似作业的角度，预测算法可以划分为分类和基于实例的学习两类。

利用分类进行预测的主要思想是使用模板来寻找与将要预测的作业相似的作业，并用这些相似的作业来预估执行时间。Downey[2]、Gibbons[9]等基于静态模板进行分类。静态模板包括用户、作业名称和系统队列等特征。分类后，用每个类的平均执行时间来预测未来作业的执行时间。Smith 等人[13]使用遗传算法动态地定义模板，即选择那些能最好的定义相似性的作业特征，并以这些特征为模板。Tsafrir 等人[6]发现只使用同一用户最近提交的两个作业也可以给出较好的预测。

基于实例的学习算法来预测作业执行时间中[1,7,8,14]，N 个最近完成的作业信息被保留，通过搜索 K 个最邻近(最类似于正在预测的作业)的作业来估计作业执行时间。该方法中的核心是如何定义两个作业之间的相似性，以及获得相似作业以后，如何利用相似作业的执行时间预测当前作业的执行时间。

作业 X 与 Y 之间的相似性由下面的距离函数给出：

$$D(x,y)=\sqrt{\frac{\sum_{a=1}^{m} w_a \times d_a(x_a,y_a)^2}{\sum_{a=1}^{m}\omega_a}} \tag{1}$$

其中：x 和 y 表示对应作业 X 和 Y 的特征向量；w_a 是权重；d_a 是相应的特征的距离。

Tran 等人[1]把分类和基于实例的学习方法结合在一起，定义一个相似度函数来度量作业的相似性，并在预测时加入了低估对预测的影响等因素。Tran 的实验结果表明其方法在大多数情况下优于单独使用某一种方法，本文即借鉴了以上混合算法并进行了部分改进。

2 作业执行时间预测

超算系统中的作业是高度重复的，用户会反复执行相同的作业，因此可以根据过去完成的作业信息来估计作业的执行时间。文献[1]指出，可以假定相似的作业有相似的执行时间，从而可以使用相似作业的执行时间给出未来作业执行时间的预测。在此前提下，预测算法分为以下两个步骤：

a) 在调度历史数据中搜索相似作业；

b) 根据相似作业的实际执行时间获得当前作业的预测时间。

在该算法中，调度历史是主要数据来源。调度系统中会将已完成执行的作业相关信息存储下来以供分析，调度历史数据一般包括作业的用户 ID、提交时间、等待时间、用户预估的执行时间、真实执行时间和所使用处理器核数等作业信息。文献[18,19]给出了一个作业历史数据的标准(standard workload format, SWF)，其中定义了作业调度相关的 18 个属性值和具体存储方式，并给出了由多个超算中心提供的标准化作业调度历史数据供研究，本文的研究工作也是以此为基础开展。

2.1 相似作业搜索

作业的相似性分为模板相似和数值相似两种。其中模板相似是指在给定模板下，判断两个作业对应属性值是否相同，属性值相同则认为两个作业为相似,模板是作业属性值的一个集合。例如，如果模板为{用户名，队列名}时，当两个作业的用户名和队列名完全相等时，本文认为这两个作业是模板相似的。本文中使用四个属性值作为模板的构成元素，即用户名、分组名、队列名、应用名，以上四个属性对应于 SWF 中的 User ID、Group ID、Queue Number、Application Number, 分别以 u、g、q、j 表示这四个属性。基于 u、g、q、j 四个属性，可以形成包括空模板在内的 16 个不同模板用于判断两个作业的相似性，在空模板下所有作业都是相似的。当然，对于不同的数据集应选择不同的模板，具体选择方法将在后面给出。

数值相似是基于作业的若干数值属性值，使用作业相似度函数，计算得到两个作业的相似度。在本文中选择三个属性值来计算数值相似度，分别为用户请求处理器数、用户请求（预估）执行时间和用户请求内存量，分别对应 SWF 中的 Requested Number of Processors、Requested Times、Requested Memory，本文中分别用 c, r, m 来表示。

在计算数值相似度之前，为了避免数值尺度不同带来的影响，本文用线性规范化函数 $f(x)$ 将各数值规范化到[0,1]区间，定义如下：

$$f(x)=\frac{x-\min_x}{\max_x-\min_x} \tag{2}$$

在完成规范化之后，使用式(3)计算作业 J_1 和 J_2 的相似度 $Sim(J_1,J_2)$。 基于 c、r、m 三个数值属性值的 $Sim(J_1,J_2)$定义如下：

$$Sim(J_1,J_2)=\sqrt{|c_1-c_2|^2+|r_1-r_2|^2+|m_1-m_2|^2} \tag{3}$$

其中：$Sim(J_1,J_2)$值越小，说明 J_1 和 J_2 的相似度越高。

本文中作业的相似性判断结合使用以上两种方法，即首先使用模板相似找出相似作业集合后再使用数值相似获得最终的相似作业集合。具体的，给定一个新作业 J，在 N 个调度历史作业中选择 K 个相似作业的步骤为：

a) 选择一个模板 T；

b) 根据模板 T，选出与 J 相似的作业形成集合 S_{J^*}；

c) 在 S_{J^*}中使用 K-近邻算法，选出 K 个与 J 相似的作业形成集合 S_J。

将调度历史作业数量限定为最临近时间完成的 N 个作业，一方面是为了提高算法效率，另一方面也是使用相近原则，即相似的作业应该是最近调度过的一些作业。算法中 N 和 K 的确定将 2.3 节讨论。

2.2 执行时间预测

在得到作业 J 的相似作业集合 S_J 后，就可以使用 S_J 中的作业的实际执行时间预测 J 的执行时间。本文中使用均值法，即将 S_J 中作业执行时间的平均值作为 J 的预测时间，计算公式如下：

$$Est(J)=\frac{\sum_{i=1}^{K}R_i}{K} \tag{4}$$

其中：R_i 是 S_J 中第 i 个作业的实际执行时间。也可以将规范化的 c、r、m 作为属性标签，采用线性回归，支持向量回归等方式获得 J 的预测时间。

为了最小化结构风险，在式(4)中加入正则化项，本文选取标准差作为正则化项，即

$$Est1(J)=Est(J)+\alpha\times std(\{R_i\}) \tag{5}$$

在文献[1]中，作者为了减少低估的可能性，加入了用户提供的执行时间，即 J 的执行时间预测公式如下：

$$Est2(J)=\min(Est1(J),\beta\times r_J) \tag{6}$$

其中：$std(.)$表示标准偏差；r_J 表示用户请求(预估) 的执行

时间； α 和 β 是加权因子。 具体取值方法将在 2.3 节中讨论。

2.3　在线参数训练

根据 2.1 和 2.2 节所示，为了获得作业 J 的执行时间，有以下几个参数需要确定：

a) 模板相似中的最佳模板；

b) 作业集的大小 N；

c) S_J 的大小 K；

d) 因子 α 和因子 β。

本文中对应的四个模板属性值用 x_1、 x_2、 x_3、 x_4 表示，若 $x_i=1$，则相应属性将被加入到模板中；反之，如果 $x_i=0$，则从模板中删除。

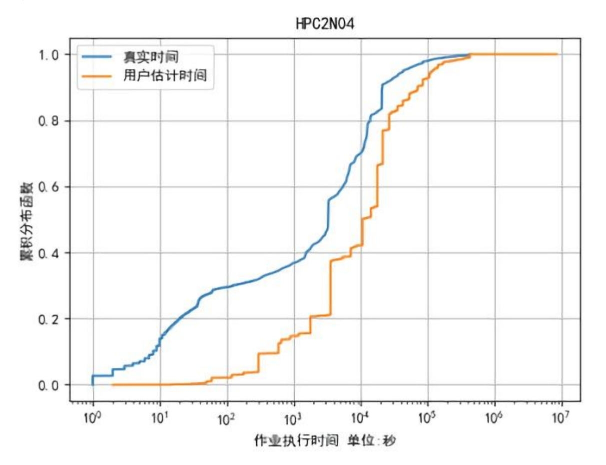
为了更好地利用超算日志的局部信息，获得最佳的预测效果，本文采用批处理的方式在线训练多组参数。每批日志分为历史数据集 HistSet (h 条作业日志)、训练集 TrainingSet (t_1 条作业日志)和测试集 TestSet (t_2 条作业日志)。历史数据集包含该批次训练中可检索的历史作业，更早提交的作业对该批次不可见。在训练集中，使用遗传算法针对参数集$(x1, x2, x3, x4, N, K, \alpha, \beta)$进行训练，训练目标是减少作业的预测误差和降低执行时间低估数量。在遗传算法中，选择如下适应度函数：

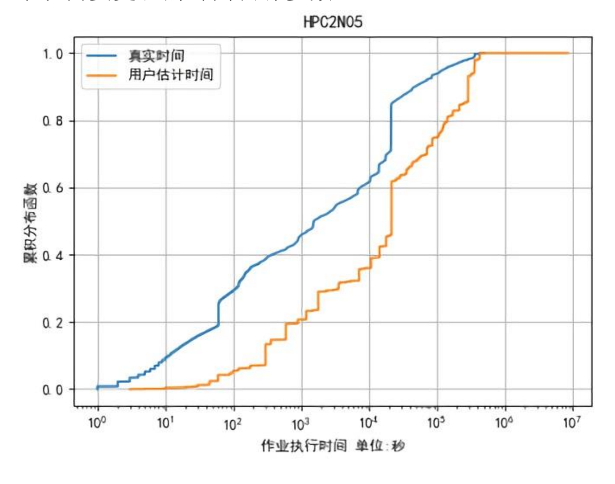
$$Fitness = \frac{\sum_i |P_i - R_i| / \sum_i R_i}{e^{(1-PU)^2}} \tag{7}$$

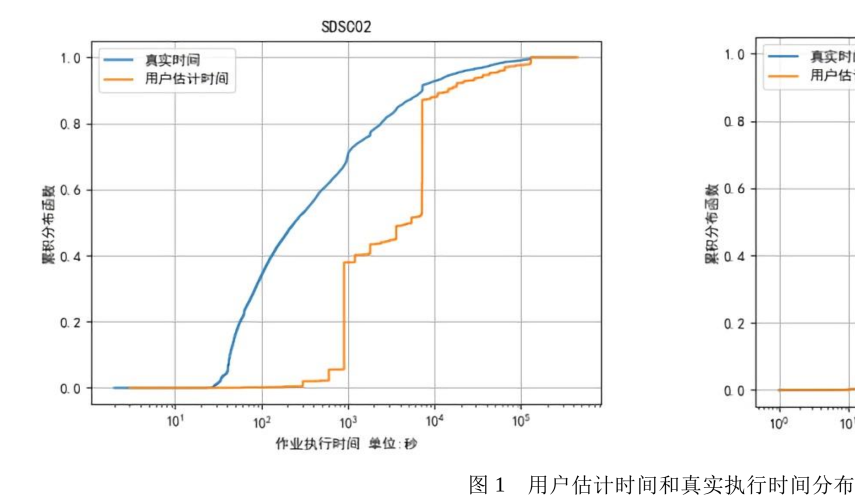
其中：P_i 和 R_i 分别是第 i 个作业对应的预测时间和实际执行时间； PU 是低估作业数占总作业数的百分比。对遗传算法设置适当的种群规模 *populations* 和迭代次数 *generations*，待算法达到收敛或次数终止即可产生一组最优的模型参数。由 2.2 节提到的执行时间预测算法给出该批次作业执行时间的预测 $\hat{r}_J$ 。

随着下一批作业的到达，HistSet、TrainingSet 和 TestSet 测试集在日志中均向后移动 t_2 条作业日志，开始下一批作业的预测。

在使用遗传算法训练作业参数时，需要对遗传算法设置相关的参数。表 1 是本文通过多轮实验选择的一组最佳的遗传算法相关参数。对比其他相关参数组合(h, t_1, t_2)如(5000, 2000, 1000)或(3000, 1500, 500)的训练结果，本文发现增大 HistSet 和 TrainingSet 的规模，或者降低每轮预测作业的数量对降低低估率和提升预测精度没有帮助。遗传算法在迭代次数达到 100 时结束训练，这是由于设置较小的迭代次数(*generations*=50)不能使作业参数收敛到全局最优。而较大的迭代次数(*generations*=200)对预测性能没有明显的改善，且系统需要更长的时间训练参数。







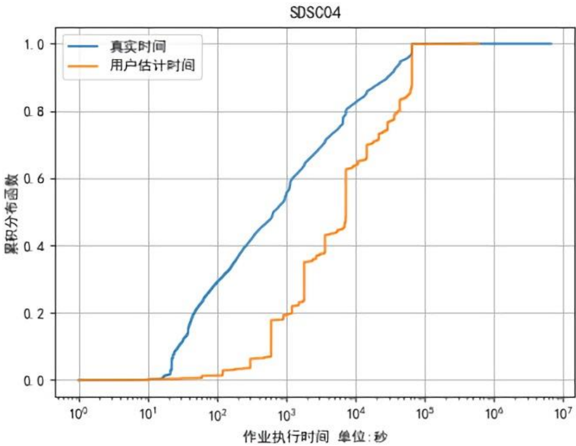


图 1　用户估计时间和真实执行时间分布

Fig. 1　Distribution of users estimated time and actual execution time

3　数值实验结果

为了方便与文献文献[1]中的结果对比(下文记为 Minh)，本文使用文献[18]中提供的北瑞典高性能计算中心的 HPC2N 日志和圣地亚哥超级计算机中心的 SDSC 日志的调度历史数据进行实验。将 HPC2N 日志按年份划分为 HPC2N04 和 HPC2N05 两个历史数据集。SDSC02 和 SDSC04 分别截取了 2002 年全年的数据和 2004 年到 2005 年的 13 个月的数据。这四个调度历史日志显示了不同集群在不同的时间段作业执行情况的细节。

表 2 显示了四个日志的基本信息。文献[1]对四个日志进行了清洗。本文保留全部有效的作业，因此处理的作业数量

比文献[1]多。

图 1 给出了四个数据集的用户估计时间和实际执行时间的概率累积分布(CDF)。该图描述了作业的预估时长和真实时长落在任一时长区间内的概率。在每个数据集上，相同的概率密度对应的真实运行时间远小于用户估计时间，这说明了用户普遍高估了作业的运行时间。由于作业执行时间变化较大，本文对时间轴做了对数处理。

从图 1 可以看出, SDSC02 和 SDSC04 的用户估计时间非常不准确，真实时间的分布较光滑而用户预估时间呈阶梯状分布，说明用户给出的预估时间比较粗略，许多用户选择了相同的估计时间。

表 1　遗传算法相关参数设置

Table 1　Parameter settings for genetic algorithm

参数名	参数值
h	3000
t_1	1500
t_2	1000
populations	50
generations	100

表 2　使用的超算日志

Table 2　Used super computing log

日志名	提交日期	作业数量	文献[1]使用的作业	本文使用的作业
HPC2N04	2004/01-2004/12	81934	81113	81934
HPC2N05	2005/01-2006/01	55389	55389	55839
SDSC02	2002/01-2002/12	100000	80756	80756
SDSC04	2004/03-2005/03	84893	66743	66743

HPC2N04 和 HPC2N05 的情况稍好一些，但作业执行时间仍然普遍被用户高估。需要注意的是，SDSC02 和 SDSC04 作业的真实执行时间一般不超过 10^5 s(约为 1 d 时间)，而 HPC2N04 和 HPC2N05 的作业真实执行时间最大不超过 10^6 s (约 12 d)，因而 HPC2N04 和 HPC2N05 的用户估计误差可能更大。

本文使用平均绝对误差 (MAE) 来度量预测的精确度，*N* 个作业的平均绝对误差计算如下：

$$MAE = \frac{\sum_i |P_i - R_i|}{N} \tag{8}$$

其中：　P_i 和 R_i 分别是第个作业对应的预测时间和实际执行时间。

对低估问题，因本文算法的目标是尽可能地减少被低估的作业数量，所以使用低估作业占所有作业的百分比来度量低估问题。作业的低估率定义如下：

$$PU = \frac{\sum_i I(P_i < R_i)}{n} \tag{9}$$

其中：　I(Pi<Ri)定义为

$$I(P_i < R_i) = \begin{cases} 1, P_i < R_i \\ 0, P_i \ge R_i \end{cases} \tag{10}$$

3.1　相似作业搜索

在采用回填策略的调度系统中，低估会导致对系统性能的不利影响。因此，预测应尽可能多地减少低估的作业数。

从表 3 显示的低估率上看，本文的预测结果与文献[1]结果相近；　同时文献中提到，将任务分为大任务和小任务能有效的降低低估率，但是本文中没有按照任务的大小进行分类，且预测的作业数更多应是导致低估率略高的原因。

减少被低估的作业数量肯定会增加高估的作业数量。然而在回填系统中，高估的作业仍然比低估更好，被高估的作业不会被系统杀死。预测器还要防止过高的估计增加预测误差，为此本文在式(6)中使用用户估计作为预测的上限，从而有效地降低训练误差。

3.2　平均绝对误差

平均绝对误差的结果见表 4。

表 3　低估率

Table 3　Percentage of underestimated jobs

数据集	Minh	本文
HPC2N04	36%	29%
HPC2N05	31%	40%
SDSC02	25%	31%
SDSC04	29%	31%

表 4　平均绝对误差(单位:秒)

Table 4　Mean absolute error (in minites)

数据集	Minh	本文	与 Minh 相比
HPC2N04	9360	5314	43%
HPC2N05	26280	6130	77%
SDSC02	1842	1839	1%
SDSC02	2886	4325	0%

从表 4 可以看出，大多数情况下本文的预测结果明显优于文献[1]的预测结果。主要原因是本文采用了不同的距离函数(式 3)。相较于文[1]中的距离函数，式（3）更能表达两个作业的相似性。 基于该距离函数，本文也在牺牲一部分低估率的前提下避免了对作业大小的分类，从而减少了遗传算法训练的参数个数。此外，本文采用在线批处理的方式按照日志到达的顺序生成预测，更好地利用了实时信息。

在 SDSC04 数据集上，本文方法预测误差相对较大。由图 1 可以看出, SDSC04 作业真实时间分布较光滑，缺乏从历史数据中学习真实执行时间的规律性。对本文的预测算法而言，具有阶梯形的真实执行时间分布的日志更容易取得良好的预测效果。

虽然本文获得的结果相对文献[1]较好，但绝对误差仍相对较大，本文基于 HPC2N04 中的部分作业，对其预测的绝对误差进行了初步分析，从而为今后算法的改进提供参考。

图 2 给出了 9 000 个作业绝对误差的累积分布图。从图中可以看到，80%的绝对误差是小于 1 000　s 的，20%左右的绝对误差相对较大，因此，今后可以分析以上 20%作业的特征，针对这些特征再进行算法的改进，以进一步降低平均绝对误差。

从图 3 可以看出，随着预测作业数的增加，平均绝对误差的基本趋势是在增加，在图中有绝对误差跃升的现象，可能出现了一些特殊的作业，这些作业的绝对误差突然增加导致平均误差的增加，今后可以通过分析这些作业的特征，通过减小这些作业的预测误差来减小总体误差。另一方面，平均绝对误差在作业数较少的时候相对较低(这在其他的作业集中也有类似现象) ，分析其原因是遗传算法只针对固定数目的历史数据进行训练，随着测试集的加入，可能需要训练新的参数去适应新的作业，这也是今后改进的方向之一。

3.3　对作业调度的影响

下面的实验通过模拟日志数据集中作业的调度与执行来量化评价本文的预测算法对作业调度的影响。实验使用开源批量调度模拟器 pyss[20] 进行基于日志的模拟。本文选择基于回填的调度策略 EASY 和 EASY-SJBF 对本节提到的四个日志数据集进行评价。通过分析不同调度策略和不同系统上的日

志数据集得到的实验结果，可以表明本文的预测算法对作业调度具有有益的影响。

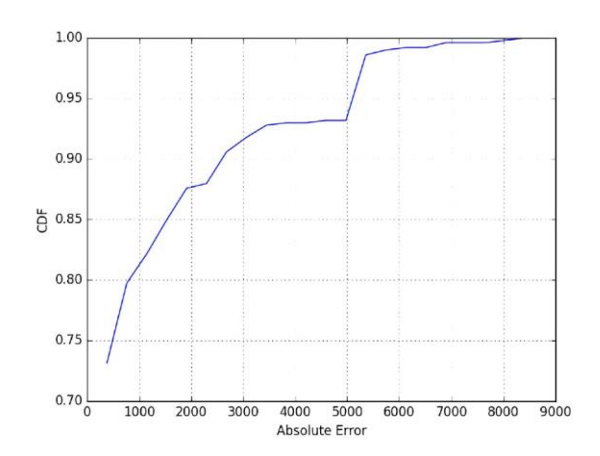


图 2　绝对误差的累积分布函数

Fig. 2　The CDF of absolute error

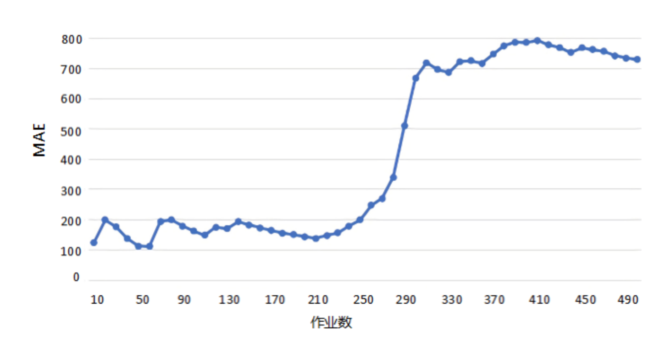


图 3　平均绝对误差随作业数变化情况

Fig. 3　MAE with different job counts

在调度策略 EASY 和 EASY-SJBF 下，回填的作业只要保证不会延迟等待队列的第一个作业的执行。EASY 按照先来先服务的原则选择可回填的作业，而 EASY-SJBF 则优先满足短作业的回填。

文献[21]指出，作业调度的性能可以用以下两个目标函数来度量:

a) 平均等待时间(*AVEwait*)。作业 j 的用户等待时间是从作业提交时刻 $submit_j$ 到作业 j 的开始时刻 $start_j$ 之间的时间段，即

$$wait_j = start_j - submit_j \tag{11}$$

等待时间越短说明作业越早开始执行，提交此作业的用户对调度越满意。所有作业的平均等待时间能够反映整体性能。

$$AVEwait = \frac{1}{n}\sum_{i=1}^{n}(start_j - submit_j) \tag{12}$$

b) 平均有界减速(*AVEbsld*)[22,23]。计算作业 j 的响应比，同时考虑作业等待时间 $wait_j$ 和作业执行时间 $runtime_j$ 两个方面。设置常量 $\tau=10s$ 避免短作业的响应比过大，不能反映调度性能。

$$bsld = \max\left(\frac{wait_j + runtime_j}{\max(runtime_j, \tau)}, 1\right) \tag{13}$$

该目标值称为作业 j 的有界减速。同样可以定义平均边界减速，降低平均有界减速可以提高系统的整体性能。

$$AVEbsld = \frac{1}{n}\sum_{i=1}^{n}\max\left(\frac{wait_j - runtime_j}{\max(runtime_j, \tau)}, 1\right) \tag{14}$$

两个目标函数从用户角度评估调度的性能。对给定的四组调度历史日志,本文首先使用用户提供的作业预估时间在 pyss 下进行调度模拟,将模拟结果的平均等待时间和平均边界减速作为基础；然后使用上节的预测结果作为预估时间进行模拟，与基础结果比较相应的目标函数值。

表 5　使用用户估计和本文算法预测时间的 *AVEwait* 比较

Table 5　Comparison of *avewait* with users estimated and the proposed prediction algorithm

日志名	调度策略	使用用户估计	使用预测时间	相对改进
HPC2N04	EASY	321.61	187.15	41%
HPC2N04	EASY-SJBF	287.27	173.57	40%
HPC2N05	EASY	723.57	439.52	40%
HPC2N05	EASY-SJBF	717.34	394.70	45%
SDSC02	EASY	188.10	98.53	48%
SDSC02	EASY-SJBF	138.24	77.08	44%
SDSC04	EASY	167.46	58.98	45%
SDSC04	EASY-SJBF	153.28	51.59	65%

表 6　使用用户估计和本文算法预测时间的 *AVEbsld* 比较

Table 6　Comparison of *avebsld* with users estimated and the proposed prediction algorithm

日志名	调度策略	使用用户估计	使用预测时间	相对改进
HPC2N04	EASY	150.24	57.02	62%
HPC2N04	EASY-SJBF	131.94	50.74	62%
HPC2N05	EASY	389.36	219.18	44%
HPC2N05	EASY-SJBF	394.05	194.44	51%
SDSC02	EASY	52.91	28.46	46%
SDSC02	EASY-SJBF	26.13	15.94	39%
SDSC04	EASY	62.63	22.52	64%
SDSC04	EASY-SJBF	46.98	15.63	67%

从表 5 和 6 的实验结果可以看出，使用本文算法的预测时间在两种调度策略和四个日志数据集上都能够显著降低平均等待时间和平均有界减速，有效地提高了调度系统的服务质量，减少了资源的空闲。

EASY-SJBF 由于选择优先回填短作业，相比 EASY 取得了更低的 *AVEwait* 和 *AVEbsld*，但两种调度策略对 *AVE-wait* 和 *AVEbsld* 的相对改进是接近的。使用预测时间对 *AV-Ebsld* 的改进更大，同时也能稳定的降低 *AVEwait*。SDSC04 是相对改进最显著的日志，在两种调度策略下 *AVEbsld* 的相对改进均超过 60%。

调度仿真的结果说明了对于采用回填的调度系统，本文的预测能够有效提高调度性能，即准确的预测有益于作业调度。

4　结束语

超算中心中作业调度的性能是影响超算系统使用率的重要因素之一，当前基于回填的调度是各超算中心主要使用的作业调度策略。在基于回填的调度中，作业的预估执行时间是能否将作业进行回填的主要判断条件，而能否精确地预测作业的执行时间会影响调度的性能。本文给出了一种结合分类和基于实例学习的在线作业性能预测算法，首先使用针对作业历史数据中的七个不同属性值进行相似作业的搜索，然后利用其中三个数值属性值计算得到估计时间。针对算法中使用到的八个参数，本文使用遗传算法进行最优值的搜索。数值实验表明，相对于文献[1]，本文使用了更少的参数获得了相近的低估率，并得到了更低的平均绝对误差。

本文对误差结果进行了初步的分析，并指出了若干改进

的方向。今后将结合机器学习，训练学习作业调度历史数据中作业的更多细节特征，以获得更好的预测效果。本文还使用预测时间进行了调度仿真，使回填算法的调度性能得到改进，下一步还要结合其他调度策略进一步探索在线执行时间预测对调度的影响。

参考文献：

[1] Tran N, Lex W. Using historical data to predict application runtimes on backfilling parallel system [C]// Proc of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing. 2010.

[2] Allen B. Predicting queue times on space-sharing parallel computers [C]// Proc of the 11th International Parallel Processing Symposium. 1997:209-218.

[3] Allen B. Using queue time predictions for processor allocation [C]//Lecture Notes in Computer Science, volume 1291. 1997:35-57.

[4] Ahuva W, Dror G. Utilization, predictability, workloads, and use runtime estimates in scheduling the IBM SP2 with backfilling [J]. IEEE Trans on Parallel and Distributed Systems, 2001,12:529-543.

[5] Dror G, Ahuva W. Utilization and predictability in scheduling the IBM SP2 with backfilling [C]// Proc of the 12th International Parallel Processing Symposium. 1998:542-546.

[6] Dan T, Yoav E, Dror G. Backfilling using system-generated predictions rather than user runtime estimates [J]. IEEE Trans on Parallel and Distributed Systems, 2007,18: 789-803.

[7] Li Hui, David L, Lex W. Mining performance data for metas-cheduling decision support in the grid [J]. Future Generation Computer Systems ,2007:92-99.

[8] Liang Feng, Liu Yunzhen, Liu Hai, *et al.* A Parallel job execution time estimation approach based on user submission patterns within computational grids [J]. International Journal of Parallel Programming, 2015, 43 (3): 440-454.

[9] Richard G. A historical application profiler for use by parallel schedulers [C]// Lecture Notes in Computer Science, volume 1291 .1997:58-77.

[10] Chiang S, Andrea A, Mary K. The impact of more accurate requested runtimes on production job scheduling performance [C]//Lecture Notes in Computer Science, volume 2537.2002:103-127.

[11] Chiang S, Mary K. Production job scheduling for parallel shared memory systems [C]// Proc of the 15th International Parallel and Distributed Processing Symposium. 2001.

[12] Vasupongayya S, Chiang S. Performance problems of using system-predicted runtimes for parallel job scheduling [C]// Proc of the 19th Parallel and Distributed Computing and Systems.2007.

[13] Smith W, Foster I, Taylor V. Predicting application run times using historical information [C]//Lecture Notes in Computer Science, volume 1459. 1998:122-142.

[14] Smith W, Wong P. Resource selection using execution and queue wait time predictions [R]. NAS Technical Report Number: NAS-02-003, NASA Ames Research Center, 2002.

[15] 余莹，李肯立，徐雨明. 计算集群中一种基于任务执行时间的组合预测方案. 计算机应用 [J] , 2015, 35 (8) :2153-2157, 2163. (Yu Ying, Li Kenli, Xu Yuming. Combined prediction scheme for runtime oftasks in computing cluster [J]. Journal of Computer Applications, 2015, 35 (8): 2153-2157, 2163.)

[16] 蒋炎华. 网格环境下任务的执行时间预测技术研究[J]. 计算机工程与设计，2011, 32 (10): 3428-3430. (Jiang Yanhua. Research of task execution time prediction technology in grid computing environments [J]. Computer Engineering and Design, 2011, 32 (10): 3428-3430.)

[17] Park J, Kim E. Runtime prediction of parallel applications with workload-aware clustering [J]. Journal of Supercomputing, 2017, 73 (3): 1-17.

[18] Dror F. Parallel workloads archive [EB/OL]. (2005-12-08) [2018-08-10]. http://www. cs. huji. ac. il/labs/parallel/workload/.

[19] Steve J, Cirne W, Dror G, *et al*. Benchmarks and standards for the evaluation of parallel job scheduler [J]. Job Scheduling Strategies for Parallel Processing. 1999:67-90.

[20] pyss-the Python scheduler simulator [EB/OL] .(2018-08-10) [2015-10-01]. http://code. google. com/p/pyss/.

[21] Dror G, Larry R, Uwe S, *et al*. Theory and practice in parallel job scheduling [C]//Lecture Notes in Computer Science.1997: 1-34

[22] Oliner A, Sahoo R, Moreira J, *et al*. Fault-aware job schedulingfor bluegene/l systems [C]// Proc of the 18th International Parallel and Distributed Processing Symposium. 2004: 64.

[23] Eric G, David G, Reis V, *et al.* Improving backfilling by using machine learning to predict running times [C]// Proc of IEEE High Performance Computing, Networking, Storage and Analysis.2017: 64.